

(一) BOM是什么

Browser Object Model 是浏览器对象模型，浏览器对象模型提供了独立与内容的、可以与浏览器窗口进行互动的对象结构，BOM由多个对象构成，其中代表浏览器窗口的window对象是BOM的顶层对象，其他对象都是该对象的子对象。

(二) BOM对象

1. window 对象，是 JS 的最顶层对象，其他的 BOM 对象都是 window 对象的属性

BOM的核心对象是window，它表示浏览器的一个实例。在浏览器中，window对象有双重角色，它既是通过javascript访问浏览器窗口的一个接口，又是ECMAScript规定的Global对象。

- 所有 JavaScript 全局对象、函数以及变量均自动成为 window 对象的成员。
- 全局变量是 window 对象的属性。
- 全局函数是 window 对象的方法。

2. document 对象，文档对象；

3. location 对象，浏览器当前URL信息；

用于获得当前页面的地址 (URL)，并把浏览器重定向到新的页面。在编写时可不使用 window 这个前缀。

```
1 location.href = 'url地址'
2     hash 返回#号后面的字符串，不包含散列，则返回空字符串。
3     host 返回服务器名称和端口号
4     pathname 返回目录和文件名。 /project/test.html
5     search 返回? 号后面的所有值。
6     port 返回URL中的指定的端口号，如URL中不包含端口号返回空字符串
7     protocol 返回页面使用的协议。 http:或https:
```

4. navigator 对象，浏览器本身信息

包含有关访问者浏览器的信息。在编写时可不使用 window 这个前缀。

```
1 navigator.platform: 操作系统类型;
2     navigator.userAgent: 浏览器设定的User-Agent字符串。
3     navigator.appName: 浏览器名称;
4     navigator.appVersion: 浏览器版本;
5     navigator.language: 浏览器设置的语言;
6     userAgent是最常用的属性，用来完成浏览器判断。
7
8     if(window.navigator.userAgent.indexOf('MSIE')!=-1){
9         alert('我是IE');
```

```
10     } else {  
11         alert('我不是IE');  
12     }
```

5. screen 对象，客户端屏幕信息

包含有关用户屏幕的信息。

```
1 screen.availWidth 属性返回访问者屏幕的宽度，以像素计，减去界面特性，比如窗口任务栏。  
2 screen.availHeight 属性返回访问者屏幕的高度，以像素计，减去界面特性，比如窗口任务栏。  
3 document.write(screen.availHeight+screen.availWidth); // 获取屏幕的宽度和高度之和
```

6. history 对象，浏览器访问历史信息

包含浏览器的历史。为了保护用户隐私，对 JavaScript 访问该对象的方法做出了限制。

```
1 history.forward() - 加载历史列表中的下一个 URL。返回下一页。  
2 history.go("baidu.com");  
3 history.forward() - 加载历史列表中的下一个 URL。返回下一页。
```

7. 页面跳转方法

```
1 window.location.href = '你所要跳转到的页面';  
2 window.open('你所要跳转到的页面');  
3 window.history.back(-1):返回上一页  
4 window.history.go(-1/1):返回上一页或下一页  
5 history.go("baidu.com");
```

1. BOM是browser object model的缩写，简称浏览器对象模型

由一系列功能的对象构成，核心对象是window，BOM缺乏标准(不过所有浏览器都支持)，JavaScript 语法的标准化组织是ECMA，DOM的标准化组织是W3C//一定要记住,BOM不是W3C的标准模型

2.window

window对象是BOM的顶层(核心)对象，所有对象都是通过它延伸出来的，也可以称为window的子对象,由于window是顶层对象，因此调用它的子对象时可以不显示的指明

//window.document.write(666)等价于document.write(666)，window下的几大功能对象(window的属性)有:navigator,screen,document,history,location

3.window的属性

```
1 closed 返回窗口是否已被关闭。
```

```
2 document    对 Document 对象的只读引用
3 history 对 History 对象的只读引用
4
5 innerHeight 返回窗口的文档显示区的高度
6 innerWidth  返回窗口的文档显示区的宽度
7 outerHeight 返回窗口的外部高度，包含工具条与滚动条
8 outerWidth  返回窗口的外部宽度，包含工具条与滚动条
9
10 screenLeft  返回相对于屏幕窗口的x坐标
11 screenTop   返回相对于屏幕窗口的y坐标
12
13 screenX  返回相对于屏幕窗口的x坐标
14 screenY  返回相对于屏幕窗口的y坐标
15
16 location    用于窗口或框架的 Location 对象
17 navigator   对 Navigator 对象的只读引用
18 onload  指定所有配置都加载完成时(图片例外)调用的函数。
19 pageXOffset 返回当前页面相对于窗口显示区左上角的 X 位置(body横向滚动的距离)
20 pageYOffset 返回当前页面相对于窗口显示区左上角的 Y 位置(body纵向滚动的距离)
21 screen  对 Screen 对象的只读引用
```

4.window的方法

```
1 alert() 显示带有一段消息和一个确认按钮的警告框。
2 close() 关闭浏览器窗口。
3 confirm() 显示带有一段消息以及确认按钮和取消按钮的对话框。
4 open(url,打开方式,新窗口配置,BOOL)打开一个新的浏览器窗口
5     4个参数都可选(一般就填第一个参数)
6     url:新窗口地址  打开方式:_blank(默认),_parent,_self,_top 配置(各种):一般默认 BOOL:新窗口
    在历史记录里面有,要不要替换
7 print() 打印当前窗口的内容。
8 prompt(tishi,value) 显示可提示用户输入的对话框。
9 scrollBy() 按照指定的像素值来滚动内容(前提是你的有滚动条:内容够多)
10 scrollTo() 把内容滚动到指定的坐标。(前提是你的有滚动条:内容够多)
11 setInterval(callback,times) 按照指定的周期(以毫秒计)来调用函数
12 setTimeout(callback,times) 在指定的毫秒数后调用函数
13 clearInterval() 取消由 setInterval() 设置的 timeout。
14 clearTimeout() 取消由 setTimeout() 方法设置的 timeout。
```

5.navigator

属性:

```
1 appCodeName 返回浏览器的代码名
2 appName 返回浏览器的名称
3 appVersion 返回浏览器的平台和版本信息
4 cookieEnabled 返回指明浏览器中是否启用 cookie 的布尔值
5 platform 返回运行浏览器的操作系统平台
6 userAgent 返回由客户机发送服务器的user-agent 头部的值
```

6.screen

属性:

```
1 availHeight 返回屏幕的高度（不包括Windows任务栏）
2 availWidth 返回屏幕的宽度（不包括Windows任务栏）
3 height 返回屏幕的总高度
4 width 返回屏幕的总宽度
5 pixelDepth 返回屏幕的颜色分辨率（每像素的位数）
```

7.history

属性:

```
1 length返回访问历史列表中的网址数
2 back() 加载 history 列表中的前一个 URL
3 forward() 加载 history 列表中的下一个 URL
4 go(number|url)加载 history列表中的某个具体页面//负数后退,正数前进
```

8.location

当前页面的url

属性:

```
1 hash 返回一个URL的锚部分//192.168.1.102:8081?name=jack&pwd=123#page1
2 host 返回一个URL的主机名和端口
3 hostname 返回URL的主机名
4 href 返回完整的URL
5 pathname 返回的URL路径名。
6 port 返回一个URL服务器使用的端口号
7 protocol 返回一个URL协议
8 search 返回一个URL的查询部分
```

方法:

```
1 assign(url) 载入一个新的文档
2 reload()    重新载入当前文档
3 replace(url) 用新的文档替换当前文档
```

弹窗:

1、alert:

alert()方法是显示一条弹出提示消息和确认按钮的警告框。

需要注意的是：alert()是一个阻塞的函数，如果我们不点确认按钮，后面的内容就不会加载出来。

```
1 alert('弹出内容');
2 console.log(123);
```

此网页显示

弹出内容

确定

2、模态框:

```
1 <style>
2     * {
3         margin: 0;
4         padding: 0;
5     }
6
7     body {
8         font-size: 14px;
9         font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
10    }
11
12    /*基类*/
13    .btn {
14        width: 200px;
15        height: 42px;
16        text-align: center;
```

```
17         line-height: 42px;
18         background-color: #000;
19         color: #fff;
20         border-radius: 5px;
21         margin: 50px auto;
22         cursor: pointer;
23         font-size: 18px;
24     }
25
26     .module {
27         position: absolute;
28         left: 0;
29         top: 0;
30
31         width: 100vw;
32         height: 100vh;
33         background-color: rgba(0, 0, 0, 0);
34
35         display: flex;
36         justify-content: center;
37         align-items: center;
38
39         /*默认隐藏: hidden表示隐藏 visible表示显示*/
40         visibility: hidden;
41     }
42
43     .module>.dialog {
44         width: 320px;
45         height: 180px;
46         border: 1px solid #666;
47         background-color: #fff;
48     }
49 </style>
50 </head>
51
52 <body>
53     <div class="btn login">登录</div>
54     <!-- 覆盖整个页面 -->
55     <div class="module">
56         <!-- 对话框 -->
```

```

57     <div class="dialog">
58         登录成功
59         <A href="javascript:void(0);" class="close">X</A>
60     </div>
61 </div>
62
63 <script>
64     //DOM 事件源
65     var login = document.querySelector('.login');
66     var close = document.querySelector('.close');
67     //要操作的盒子
68     var module = document.querySelector('.module');
69     //事件三要素
70     login.onclick = function () {
71         //把隐藏的模态框显示出来
72         module.style.visibility = 'visible';
73     }
74
75     close.onclick = function () {
76         //把模态框隐藏
77         module.style.visibility = 'hidden';
78     }
79 </script>

```

案例:

```

1
2 <style>
3     *{
4         padding: 0;
5         margin:0;
6     }
7     body{
8         width: 100%;
9         height:100%;
10        background: #fff;
11    }
12    .toggleModal{
13        padding: 10px 20px;

```

```
14         color: white;
15         background: #409EFF;
16         border:none;
17         box-shadow: 2px 3px 20px rgba(0,0,0,0.2);
18         position: absolute;
19         left: 50%;
20         top: 50%;
21         transform: translate(-50%,-50%);
22     }
23     .mask{
24         position: absolute;
25         top: 0;
26         bottom: 0;
27         left: 0;
28         width: 100%;
29         height: 100%;
30         background: rgba(0,0,0,0.4);
31         z-index: 100;
32         display: none;
33     }
34     .modal{
35         position: absolute;
36         top: 50%;
37         left: 50%;
38         width: 400px;
39         transform: translate(-50%,-50%);
40         border-radius: 5px;
41         background: #fff;
42         box-shadow: 2px 3px 20px rgba(0,0,0,0.2);
43         z-index: 120;
44         display: none;
45     }
46     .modal .modal-header{
47         height: 50px;
48         border-bottom: 1px solid #f5f5f5;
49         padding: 0 15px;
50     }
51     .modal .modal-header p {
52         line-height: 50px;
53         display: inline-block;
```



```
54     }
55     .modal .modal-header .title{
56         font-size: 18px;
57         color: #333;
58     }
59     .modal .modal-header .close{
60         float: right;
61         font-size: 26px;
62         margin-top: -2px;
63         color: #9C9FA4;
64         cursor: default;
65     }
66     .modal .modal-content{
67         min-height: 100px;
68     }
69     .modal .modal-footer .btn{
70         padding: 0 20px;
71         height: 36px;
72         line-height: 36px;
73         color: white;
74         background: #409EFF;
75         border: none;
76     }
77     .modal .modal-footer{
78         border-top: 1px solid #f5f5f5;
79         padding: 15px;
80         text-align: right;
81     }
82     .container::after{
83         content:"";
84         display: block;
85         clear: both;
86     }
87 }
88 </style>
89 </head>
90 <body>
91 <div class="container">
92     <div>
```

```
93         <button class="toggleModal" id="toggleModal">触发遮罩层</button>
94     </div>
95     <div class="modal">
96         <div class="modal-header">
97             <p class="title">这是弹出层的标题</p>
98             <p class="close">×</p>
99         </div>
100        <div class="modal-content">
101
102        </div>
103        <div class="modal-footer">
104            <button class="close btn">关闭</button>
105        </div>
106    </div>
107    <div class="mask"></div>
108 </div>
109 <script>
110     window.onload = function(){
111         // 获取需要使用到的元素
112         var toggleModal = document.getElementById("toggleModal");
113         var container = document.getElementsByClassName("container")[0];
114         var mask = document.getElementsByClassName("mask")[0];
115         var modal = document.getElementsByClassName("modal")[0];
116         var closes = document.getElementsByClassName("close");
117         toggleModal.onclick = show;
118         closes[0].onclick = close;
119         closes[1].onclick = close;
120         function show(){
121             mask.style.display = "block";
122             modal.style.display = "block";
123         }
124         function close(){
125             mask.style.display = "none";
126             modal.style.display = "none";
127         }
128     }
129 </script>
```

触发遮罩层

3、prompt:

prompt()方法是显示提示用户进行输入的对话框。

这个方法返回的是用户输入的字符串。

```
1 var age = prompt('请输入年龄: ');  
2 console.log(age);
```

此网页显示

请输入年龄:

确定

取消

4、confirm:

confirm()方法是显示一个含有指定消息和确认和取消按钮的确认框。

如果点击"确定"返回true，否则返回false。

```
1 var submit = confirm('是否确定删除? ');  
2 console.log(submit);
```

此网页显示

是否确定删除?

确定

取消

2 BOM: 定时器

定时器setTimeout:

setTimeout() 方法用于在指定的毫秒数后调用函数或计算表达式；在指定时间内, 将任务放入事件队列,等待js引擎空闲后被执行

语法:

setTimeout(a:函数表达式, b:毫秒数, c:参数);

a: 必需的参数, 表示需要调用的函数或要执行的代码串。

b: 必须的参数, 表示周期性执行或调用 “a参数” 前的时间间隔, 以毫秒为单位计时 (1s=1000ms)。

c: 可选的参数。

setTimeout()只执行函数一次, 如果需要多次调用可以使用setInterval(),或者在函数体内再次调用setTimeout()

```
1 // setTimeout(函数, 时间, 参数),
2     console.log(1);
3     for (var i = 0; i <= 5; i++) {
4         setTimeout(function () {
5             //用同步的代码创建异步的任务
6             console.log(i);
7         }, 0)
8     }
9     console.log(3);
10 //结果是1-->3-->6个6, setTimeout(0)并不是立即执行
```

setTimeout(0)的作用:

HTML5定义的最小时间间隔是4毫秒., 不同浏览器的实现情况也不同, 使用setTimeout(0)会使用浏览器支持的最小时间间隔。所以当我们需把一些操作放到下一帧处理的时候,我们通常使用setTimeout(0)来hack.

js引擎是单线程执行的:

js是没有多线程的.js引擎的执行是单线程执行。

```
1 <script>
2     var start = true;
3     setTimeout(function () {
4         start = false;//1s后, 改变start的值
5     }, 1000);
6     while (start);
```

```
7     console.log('end');
8 </script>
```

start默认是true,在while中是死循环的。最后的console.log是不会执行的。添加一个定时器，1秒后将start改为false。如果说js引擎是多线程的，那么在1秒后，console.log就会被执行。但是页面会永远死循环下去，console.log并没有执行。这证明setTimeout并不能作为多线程使用，因此js引擎执行是单线程的。

定时器传参:

```
1 // setTimeout(), 定时器传参
2     for (var i = 0; i <= 5; i++) {
3         setTimeout(function (n) {
4             console.log(n);
5         }, 0, i);
6     }
```

setInterval:

setInterval()方法可按照指定的周期来调用函数或者计算表达式（以毫秒为单位）

语法:

setInterval(函数表达式, 毫秒数, 参数);

setInterval()会不停的调用函数，直到clearInterval()被调用或者窗口被关闭，由 setInterval()返回的ID值可用作clearInterval()方法的参数

```
1 //setInterval
2     var count = 0;
3     var sid = setInterval(function () {
4         console.log(100);
5         //count++ === 5 && clearInterval(sid); //6个100
6         ++count === 5 && clearInterval(sid);
7     }, 3000);
```

clearTimeout(): 停止定时器

要使用clearTimeout(), 需要我们设定setTimeout()时，给予这setTimeout()一个名称

clearInterval(): 停止定时器

要使用clearInterval(), 需要我们设定clearInterval()时，给予这clearInterval()一个名称

```
1 //setTimeout 1000ms后执行1次
```

```

2  var test1 = setTimeout(function(){
3    },1000);
4
5  //setInterval 每隔1000ms执行一次
6  var test2 = setInterval(function(){
7
8    },1000)
9
10 //清除Timeout的定时器,传入变量名(创建Timeout定时器时定义的变量名)
11 clearTimeout(test1);
12
13 //清除Interval的定时器,传入变量名(创建Interval定时器时定义的变量名)
14 clearInterval(test2);

```

<https://www.h5anli.com/articles/201705/setimeone.html>

案例:

```

1  * {
2      margin: 0;
3      padding: 0;
4  }
5
6  body {
7      font-size: 14px;
8      font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
9  }
10 li{
11     list-style: none;
12 }
13 /*基类*/
14 .btn {
15     width: 200px;
16     height: 42px;
17     text-align: center;
18     line-height: 42px;
19     background-color: #000;
20     color: #fff;
21     border-radius: 5px;
22     margin: 50px auto;

```

```
23         cursor: pointer;
24         font-size: 18px;
25     }
26
27     .ball{
28         display: flex;
29         justify-content: space-between;
30         width: 1200px;
31         margin: auto;
32     }
33     .ball > li{
34         width: 120px;
35         height: 120px;
36         border: 2px solid red;
37         text-align: center;
38         line-height: 120px;
39         font-weight: bold;
40         font-size: 42px;
41         color: red;
42         border-radius: 50%;
43     }
44     .ball > li:last-child{
45         color: blue;
46         border-color: blue;
47     }
48
49 </style>
50
51 </head>
52
53 <body>
54     <ul class="ball">
55         <li>1</li>
56         <li>2</li>
57         <li>3</li>
58         <li>4</li>
59         <li>5</li>
60         <li>6</li>
61         <li>7</li>
```

```

62     </ul>
63     <div class="btn start">开始摇奖</div>
64     <script>
65         /**
66             1, 6个红球，取值范围是1~33;
67             2, 红球不能重复;
68             3, 一个蓝球，取值范围是1~16;
69         */
70         var start = document.querySelector('.start');
71         //拿到所有的li，是一个类数组
72         var lisArr = document.querySelectorAll('.ball>li');
73         /** 生成指定范围内的一个随机的整数*/
74         rand(20, 78);
75         function rand(min, max){
76             return Math.round(min + Math.random() * (max-min));
77         }
78         start.onclick = function(){
79             var red = Array.from(new Array(34).keys());
80             red.shift();//删除第一个元素 并返回第一个元素
81             var blue = red.slice(0, 16);//截取数组，蓝色的球用的数字
82             var count = 0;//计数器
83             var ballIndex = 0;
84             var sid = setInterval(function(){
85                 var ind= rand(0, red.length-1);
86                 lisArr[ballIndex].innerHTML = red[ind];
87                 //把使用过的球删除
88                 ++count % 15 || (ballIndex++ && red.splice(ind, 1));
89                 //进入蓝色的球
90                 if(ballIndex === 6) red = blue;
91                 //清除定时器
92                 if(ballIndex === 7) clearInterval(sid);
93
94             }, 87);
95         }
96     </script>

```

秒杀倒计时:

案例1


```
1 <style>
2     * {
3         margin: 0;
4         padding: 0;
5     }
6
7     body {
8         font-size: 14px;
9         font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
10    }
11
12    li {
13        list-style: none;
14        display: inline-block;
15    }
16
17    .djs {
18        width: 280px;
19        height: 50px;
20        background-color: red;
21        padding: 5px 58px;
22        /*怪异盒子*/
23        box-sizing: border-box;
24        margin: auto;
25    }
26
27    .djs>li:nth-child(2n+1) {
28        width: 45px;
29        height: 45px;
30        background-color: #333;
31        color: #fff;
32        text-align: center;
33        line-height: 45px;
34        font-weight: bolder;
35        font-size: 24px;
36
37    }
38
```

```

39     .djs>li:nth-child(2n) {
40         color: #fff;
41         font-weight: bolder;
42         font-size: 22px;
43     }
44 </style>
45 </head>
46
47 <body>
48     <ul class="djs">
49         <li>02</li>
50         <li>:</li>
51         <li>25</li>
52         <li>:</li>
53         <li>33</li>
54     </ul>
55     <script>
56         //结束时间
57         var end = new Date('2022-05-11 21:30:00');
58         //提前把要操作的标签拿过来
59         var lisArr= document.querySelectorAll('.djs>li');
60         setInterval(function () {
61             var last = Math.trunc((end - new Date()) / 1000); //隐式类型转换
62             //相差的小时数
63             var hours = Math.trunc(last / 3600);
64             //相差的分钟
65             var minute = Math.trunc(last / 60) % 60;
66             //相差的秒数
67             var second = last % 60;
68
69             lisArr[0].innerHTML = hours.toString().padStart(2, '0');
70             lisArr[2].innerHTML = minute.toString().padStart(2, '0');
71             lisArr[4].innerHTML = second.toString().padStart(2, '0');
72         }, 1000);
73     </script>
74

```

案例2

```
1 <style>
2     * {
3         margin: 0;
4         padding: 0;
5     }
6
7     body {
8         color: #fff;
9     }
10
11     .box {
12         margin: 20px auto;
13         position: relative;
14         width: 235px;
15         height: 322px;
16         background-color: #e62d20;
17         background-image: url(./image/jd.png);
18         background-size: contain;
19         background-position: 50%;
20     }
21
22     .box .title {
23         padding-top: 30px;
24         float: top;
25         text-align: center;
26         color: #fff;
27         font: 40px 'Microsoft YaHei';
28         font-weight: 700px;
29         width: 100%;
30         height: 45px;
31         line-height: 45px;
32     }
33
34     .words {
35         text-align: center;
36         display: inline-block;
37         padding-top: 125px;
38         height: 26px;
```

```
39         width: 100%;
40         line-height: 26px;
41     }
42
43     .box .shijian {
44         width: 100%;
45         height: 50px;
46         margin-top: 10px;
47         display: block;
48     }
49
50     .hour {
51         margin-left: 32px;
52     }
53
54     .hour,
55     .minute,
56     .second {
57         display: inline-block;
58         width: 50px;
59         height: 50px;
60         line-height: 50px;
61         text-align: center;
62         background-color: #2f3430;
63         color: white;
64         font-size: 25px;
65         font-weight: 800px;
66     }
67
68     .m1,
69     .m2 {
70         color: #fff;
71         font-size: 25px;
72         font-weight: 800px;
73     }
74 </style>
75 </head>
76
77 <body>
78     <div>
```

```

79     <div class="box">
80         <div class="title">京东秒杀</div>
81         <span class="words"><strong> 20:00</strong>点场 距结束</span>
82         <span class="shijian">
83             <span class="hour"></span>
84             <span class="m1">:</span>
85             <span class="minute"></span>
86             <span class="m2">:</span>
87             <span class="second"></span>
88         </span>
89     </div>
90 </div>
91
92 <script>
93     var hour = document.querySelector(".hour");
94     var minute = document.querySelector(".minute");
95     var second = document.querySelector(".second");
96     var inputTime = +new Date("2022-5-11 20:00:00"); //倒计时的结束时间，自己设置时间
97     countdown(); //先调用一次这个函数 防止第一次刷新页面有空白
98     //2、开启定时器
99     setInterval(countDown, 1000); //1000毫秒，每一秒钟调用一次函数
100    //3、倒计时-时分秒函数
101    function countDown() {
102        var nowTime = +new Date(); //返回的是当前时间的总的毫秒数
103        var times = (inputTime - nowTime) / 1000; // times是剩余时间的总的毫秒数
104        var h = parseInt(times / 60 / 60 % 24);
105        h = h < 10 ? '0' + h : h; //判断数值小于10的情况 如 0-9改为 00-09
106        hour.innerHTML = h; //更改div里面的内容 把h给获取元素hour的内容
107
108        var m = parseInt(times / 60 % 60);
109        m = m < 10 ? "0" + m : m;
110        minute.innerHTML = m; //同上
111
112        var s = parseInt(times % 60);
113        s = s < 10 ? "0" + s : s;
114        second.innerHTML = s; //同上
115    }
116 </script>

```

跳转:

```
1  setTimeout(function () {
2      window.location.href = './1.秒杀倒计时.html';
3  }, 3000);
```

登陆数据处理页面:

```
1      <title>数据处理</title>
2  </head>
3
4  <body>
5      <script>
6          console.log(window.location);
7          /**自己扩展一个方法用于接收get请求提交过来的数据*/
8          window.location.querystring = function () {
9              /** 1,window.location.search: 查询字符串 "?username=chenwen&passwd=123456"
10                 2,window.location.search.substring(1):  "username=chenwen&passwd=123456"
11                 3,...substring(1).split('&'):  ['username=chenwen', 'passwd=123456']
12                 4, ...split('&').forEach: 对切割后的数组进行遍历
13                 */
14
15                 var obj = {};
16                 //this 就是当前对象window.location
17                 this.search.substring(1).split('&').forEach(function (q) {
18                     //q.split('='); //'username=chenwen' ===> ['username', 'chenwen']
19                     //向obj追加属性(切割后的数组的第1个元素)并赋值(切割后的数组的第2个元素)
20                     obj[q.split('=').shift()] = q.split('=').pop();
21                 });
22                 return obj;
23             }
24
25             //querystring 这个方法是自己定义的
26             var formdata = window.location.querystring();
27             console.log(formdata); //{username:'chenwen', passwd:'123456'}
28
29             //对账号和密码进行验证
30             var userList = { //模拟的用户数据--》真正的情况是数据库里面的数据
31                 zhansan: '123456',
32                 lisi: '234567',
```

```

33         wanger: '345678',
34         mazi: '456789'
35     };
36     //hasOwnProperty 和 in 不一样
37     if (!userList.hasOwnProperty(formdata.username)) {
38         // !userList[formdata.username]
39         alert('账号不存在');
40         window.location.href = './1.html';
41     } else if (formdata.password !== userList[formdata.username]) {
42         alert('密码错误');
43         window.location.href = './1.html';
44     } else {
45         alert('登陆成功');
46         window.location.href = './1.html';
47     }
48 </script>

```

定时器案例: https://blog.csdn.net/weixin_34332905/article/details/88752311?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522165219458016782184620990%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=165219458016782184620990&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-1-88752311-null-null.142^v9^pc_search_result_cache,157^v4^control&utm_term=%E5%AE%9A%E6%97%B6%E5%99%A8%E6%A1%88%E4%BE%8B&spm=1018.2226.3001.4187